University of Wollongong

Research Online

Faculty of Engineering and Information Sciences - Papers: Part A

Faculty of Engineering and Information Sciences

1-1-2020

Using cost-sensitive learning and feature selection algorithms to improve the performance of imbalanced classification

Fang Feng

Kuan-Ching Li

Jun Shen
University of Wollongong, jshen@uow.edu.au

Qingguo Zhou
zhouqg@lzu.edu.cn

Xuhui Yang

Follow this and additional works at: https://ro.uow.edu.au/eispapers

 Part of the Engineering Commons, and the Science and Technology Studies Commons

Recommended Citation

Feng, Fang; Li, Kuan-Ching; Shen, Jun; Zhou, Qingguo; and Yang, Xuhui, "Using cost-sensitive learning and feature selection algorithms to improve the performance of imbalanced classification" (2020). *Faculty of Engineering and Information Sciences - Papers: Part A*. 6783.
https://ro.uow.edu.au/eispapers/6783

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Using cost-sensitive learning and feature selection algorithms to improve the performance of imbalanced classification

Abstract

Imbalanced data problem is widely present in network intrusion detection, spam filtering, biomedical engineering, finance, science, being a challenge in many real-life data-intensive applications. Classifier bias occurs when traditional classification algorithms are used to deal with imbalanced data. As already known, the General Vector Machine (GVM) algorithm has good generalization ability, though it does not work well for the imbalanced classification. Additionally, the state-of-the-art Binary Ant Lion Optimizer (BALO) algorithm has high exploitability and fast convergence rate. Based on these facts, we have proposed in this paper a Cost-sensitive Feature selection General Vector Machine (CFGVM) algorithm based on GVM and BALO algorithms to tackle the imbalanced classification problem, delivering different cost weights to different classes of samples. In our method, the BALO algorithm determines the cost weights and extract more significant features to improve the classification performance. Experiments conducted on eleven imbalanced data sets have shown that the CFGVM algorithm significantly improves the classification performance of minority class samples. By comparing with similar algorithms and state-of-the-art algorithms, the proposed algorithm significantly outperforms in performance and produces better classification results.

Keywords

algorithms, selection, feature, classification, learning, imbalanced, cost-sensitive, performance, improve

Disciplines

Engineering | Science and Technology Studies

Publication Details

Feng, F., Li, K., Shen, J., Zhou, Q. & Yang, X. (2020). Using cost-sensitive learning and feature selection algorithms to improve the performance of imbalanced classification. IEEE Access, 8 (1), 69979-69996.

This journal article is available at Research Online: https://ro.uow.edu.au/eispapers/6783

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 7, 2020, accepted April 7, 2020, date of publication April 13, 2020, date of current version April 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2987364

Using Cost-Sensitive Learning and Feature Selection Algorithms to Improve the Performance of Imbalanced Classification

FANG FENG[1,2], KUAN-CHING LI[3], (Senior Member, IEEE), JUN SHEN[4,5], QINGGUO ZHOU[1], AND XUHUI YANG[1]

[1] School of Information Science and Engineering, Lanzhou University, Lanzhou 730000, China
[2]School of Electronic and Information Engineering, Lanzhou Institute of Technology, Lanzhou 730050, China
[3]Department of Computer Science and Information Engineering, Providence University, Taichung 43301, Taiwan
[4]School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia
[5]Department of EE and CS, Research Lab of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Corresponding author: Qingguo Zhou (zhouqg@lzu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61402210 and Grant 61872079, in part by the Fundamental Research Funds for the Central Universities under Grant lzujbky-2019-kb51 and Grant lzujbky-2018-k12, in part by the Ministry of Education-China Mobile Research Foundation under Grant MCM20170206, in part by the Major National Project of High Resolution Earth Observation System under Grant 30-Y20A34-9010-15/17, in part by the State Grid Corporation Science and Technology Project under Grant SGGSKY00FJJS1800403 and Grant 522722160071, in part by the Program for New Century Excellent Talents in University under Grant NCET-12-0250, in part by the Double First Class Funding-International Cooperation and Exchange Program under Grant 227000-560001, in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA03030100, and in part by the Google Research Awards and Google Faculty Award, University of Wollongong's UGPN RCF 2018–2019 and UIC Scheme EIS Near Miss.

ABSTRACT Imbalanced data problem is widely present in network intrusion detection, spam filtering, biomedical engineering, finance, science, being a challenge in many real-life data-intensive applications. Classifier bias occurs when traditional classification algorithms are used to deal with imbalanced data. As already known, the General Vector Machine (GVM) algorithm has good generalization ability, though it does not work well for the imbalanced classification. Additionally, the state-of-the-art Binary Ant Lion Optimizer (BALO) algorithm has high exploitability and fast convergence rate. Based on these facts, we have proposed in this paper a Cost-sensitive Feature selection General Vector Machine (CFGVM) algorithm based on GVM and BALO algorithms to tackle the imbalanced classification problem, delivering different cost weights to different classes of samples. In our method, the BALO algorithm determines the cost weights and extract more significant features to improve the classification performance. Experiments conducted on eleven imbalanced data sets have shown that the CFGVM algorithm significantly improves the classification performance of minority class samples. By comparing with similar algorithms and state-of-the-art algorithms, the proposed algorithm significantly outperforms in performance and produces better classification results.

INDEX TERMS Imbalanced data, cost-sensitive, general vector machine, binary ant lion optimizer.

I. INTRODUCTION

In traditional classification research, there are some basic assumptions: (1) The numbers of samples approximately equal across different classes; (2) The misclassification cost in different classes is also roughly the same. However, in practical applications, the above two assumptions are difficult to hold. The distribution of raw data in many applications is imbalanced, as they focus on minority categories of related information, such as detecting illegal transactions in credit cards, mining coding information in gene sequences, and detecting oil pollution on the sea surface. In the detection of illegal credit card transactions, most of these credit card transactions are standard transactions, and only a small number of credit card transactions are unlawful, which is due to the probability of events leading to data imbalance.

Data collection of some application-specific classes is delicate due to price and privacy issues. If the traditional classification algorithm is applied to classify the data of the imbalance problem, the overall recognition rate can be

The associate editor coordinating the review of this manuscript and approving it for publication was Juan Wang.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

excellent, though the recognition rate of the minority classes that we focus on may not be reasonable, which is due to the conventional classification algorithm treating all categories of data equally, ignoring the relative distribution of different groups, resulting in biased towards the majority class.

For imbalance classification problems, we aim at improving the recognition rate of minority classes as much as possible, while not reducing the recognition rate of majority classes. The main reason is that the recognition rate of minority classes is often more important than that of majority classes. For instance, in a network intrusion problem, if the intrusion behavior is misclassified into normal behavior, it will lead to incalculable losses, and the normal behavior being misclassified into intrusion behavior will not cause significant impact [1]. In cancer detection, if cancer patients are misclassified as normal patients, they miss the best treatment time and eventually lead to death. The misclassification of normal patients into cancer patients may lead to mental tension, which will not cause fatal harm to the former [2], though. The inefficiency of traditional classification algorithms and the importance of the recognition rate of minority classes inspire us to find new ways to solve the class imbalance problem.

This paper proposes a hybrid algorithm CFGVM based on a cost-sensitive learning algorithm and feature selection algorithm to solve the class imbalance problem. The contributions are listed below:

1) A CFGVM algorithm is proposed based on the GVM algorithm and the BALO algorithm. The GVM algorithm is a new classification algorithm proposed by Zhao in 2016 [3]. The BALO algorithm is a new nature-inspired algorithm proposed by E. Emary in 2016 [4]. To the best of our knowledge, our proposed method is the first research work aiming at utilizing the GVM algorithm and the BALO algorithm to solve the imbalanced classification problem. Firstly, The BALO algorithm is used to improve the GVM algorithm to a Cost-sensitive General Vector Machine (CGVM) algorithm by assigning differential misclassification costs to different classes, and then the BALO algorithm is used to search the optimal feature subset to further improve the classification performance to a CFGVM algorithm.

2) Experiments are carried out on on eleven datasets. In comparison with state-of-the-art algorithms, the effectiveness of the proposed algorithm has been verified. In all the comparison algorithms, Six evaluation indicators of CFGVM1 algorithm can obtain the best results on eight datasets, Six evaluation indicators of CFGVM2 algorithm can obtain the best results on nine datasets, while six evaluation indicators of CFGVM3 algorithm can obtain the best results on all the eleven datasets. It illustrates that *G-mean* can slightly better evaluate the classification performance of imbalanced classifications compared to *F-measure* and *Accuracy* in this paper.

The rest of this paper is organized as follows: Section 2 outlined the related work of cost-sensitive learning, Ant Lion Optimizer (ALO) and GVM. Section 3 gives the details of the proposed algorithm. The experimental results are discussed in Section 4, and finally, conclusion and future work are summarized in Section 5.

TABLE 1. Cost matrix for binary problems.

Project	Predicted as category 1	Predicted as category 2
True for the first category	C_{11}	C_{12}
True for the second category	C_{21}	C_{22}

II. RELATED WORK

A. COST-SENSITIVE LEARNING ALGORITHM

The cost-sensitive learning algorithms improve the classification performance of imbalanced classification by assigning different misclassification costs to the minority and the majority class samples. The key to the cost-sensitive learning algorithm is to determine the optimal value of the costs by minimizing the total misclassification cost in the training samples [5]. The cost matrix of the binary classification is shown in Table 1. C_{ij} represents the cost of predicting a sample of class i as class j. When $i = j$, which means the cost for correct classification is 0. In the cost-sensitive algorithm, $C_{ij} \neq C_{ji}$. Generally speaking, minority classes are more important than majority classes [6], so in practice, we should give greater punishment to minority classes. The design of the cost matrix is related to the final classification. Suppose category i is a minority class. For simplicity, we can set C_{ji} to a constant 1 and then find the optimal value of C_{ij}. The binary imbalanced classification problem is of particular interest in this paper.

In the past decades, many cost-sensitive learning algorithms have been proposed to deal with the class imbalance problem, which could be divided into six categorites depending on different strategies. The first category is to modify the objective function using a weighting strategy. Cheng *et al.* [7] proposed a cost-sensitive large margin distribution machine to obtain a balanced classifier, which introduced the cost-sensitive margin mean and minimizes the cost-sensitive penalty. Wu *et al.* [8] proposed a mixed-kernel based weighted extreme learning machine considering the influence of kernel function on human activity recognition, which results showed that the proposed algorithm outperforms extreme learning machine (ELM) and weighted ELM algorithm on *Accuracy*, *G-mean* metrics in the UCI dataset. Phoungphol *et al.* [9] proposed a new multiclass Support Vector Machine (SVM) algorithm for imbalance learning. The algorithm formalized a new objective function to maximize *G-mean*, which results showed that the proposed algorithm could be effectively applied in multiclass imbalance problem. Maldonado and Weber [6] proposed a family of embedded algorithms for backward feature selection using a cost-sensitive metric that becomes flexible. Compared with the well-known feature selection algorithm, the proposed algorithm could acquire better classification performance with fewer features on the six imbalanced data sets.

The second class of algorithms involves tree-building strategies. Del Rio *et al.* [10] proposed a new ensemble creation algorithm that incorporates Random Balance to AdaBoost.M2. The novelty of the algorithm is that the proportions of each member in the ensemble algorithm are randomly selected. A cost-sensitive ensemble was proposed by Krawczyk *et al.* [11], in which authors combined random subspace based feature space with cost-sensitive decision trees. Sahin *et al.* [12] developed a new algorithm to select the splitting attribute at each non-terminal node by minimizing the total misclassification costs.

The third class of algorithms is to introduce a cost factor into the fuzzy rule-based classification systems. López *et al.* [13] presented a new algorithm Chi-FRBCS-BigDataCS based on a linguistic cost-sensitive fuzzzy rule-based algorithm for imbalanced learning. Vluymans *et al.* [14] developed a framework to solve the multi-instance class imbalance problem based on the fuzzy rough set theory, which was composed of the proposed two types of the classifier. The class-dependent weight vectors were used in ordered weighted aggregation.

The fourth class of algorithms is to use a cost-sensitive error function on the neural network. Oh *et al.* [15] proposed a new scheme based on Active Example Selection (AES). Different from most traditional machine learning algorithms, AES started learning with a small subset of size and increased gradually. Therefore, the objective functions of AES are also different from the conventional activation function. A new algorithm cost-sensitive multilayer perceptron based on a joint objective function is proposed for imbalanced classification [16]. The importance of class errors is distinguished by using a single cost parameter. Ghazikhani *et al.* [17] presented two online classifiers to solve the problem of concept drift and class imbalance that incorporated two different cost-sensitive strategies to the objective function of the online one-layer Neural Network.

The fifth class of algorithms is cost-sensitive boosting algorithms. Sun *et al.* [18] studied cost-sensitive boosting algorithms to cope with the class imbalance problem. The proposed algorithm included a new weight updating strategy by weighing each sample based on its associated cost item. Wang *et al.* [19] proposed a new boosting-SVMs to deal with the class imbalance problem. The proposed algorithm combined the resampling algorithm with a cost-sensitive learning algorithm.

The sixth class of algorithms is based on Bayes decision theory. Ali *et al.* [20] developed an efficient ensemble system Cost-Sensitive Classifier with GentleBoost Ensemble, which integrated cost sensitive learning into GentleBoost, AdaBoostM1, and Bagging. GentleBoost used Decision Tree as base learners. Datta and Das [21] proposed an improved support vector machine algorithm by minimizing the overall Bayes error with equal misclassification costs and target class misclassification with unequal costs. Bahnsen *et al.* [22] proposed a cost measure to evaluate a cost-sensitive learning algorithm based on Bayes minimum risk.

In addition to cost-sensitive learning algorithms, some researchers present other algorithms to deal with the class imbalanced problem. This paper is highly related to cost-sensitive learning algorithms, so other algorithms are only briefly introduced. Kang *et al.* [23] proposed a Noise-filtered Under-sampling Scheme based on under-sampling and noise filtering for imbalanced learning. Compared with the original undersampling-based algorithms, the proposed scheme could acquire better classification performance in terms of *F-measure*, *G-mean* and the area under the curve. Kang *et al.* [24] presented a weighted undersampling scheme for SVM based on space geometry distance to cope with the class imbalanced problem. Compared with the state-of-the-art algorithms, the proposed algorithm could acquire better classification performance on the 27 imbalanced data sets. Liu *et al.* [25] developed an embedded feature selection algorithm named weighted Gini index by adding an index-weighting method to classification and regression tree to deal with the class imbalanced problem. Experiments shown that WGI could achieve the better performance only if 20% or more of features are chosen compared to Chi2, F-statistic and Gini index. Tang *et al.* [26] construct a comprehensive feature vector and develop an ensemble identification algorithm to deal with real-word Weibo datasets, which results showed that the recall rate of the proposed algorithm increased by 6.5% compared with the existing state-of-the-art algorithms.

B. ALO AND GVM

ALO is a new metaheuristic optimization algorithm, which has the advantage of high development and convergence [27]. Mirjalili has successfully resolved the three typical engineering problems by using the ALO [27]. After that, E. Emary proposed the BALO based on the ALO for feature selection [4]. Experiments have shown that the proposed BALO is better than particle swarm optimizer (PSO), genetic algorithms (GAs), binary bat algorithm (BBA), and the ALO on 21 data sets. Seyedali Mirjalili proposed a Multi-Objective Ant Lion Optimizer to solve the multi-objective optimization problem, which results showed that this algorithm outperformed Non-dominated Sorting Genetic Algorithm II on the majority of the test functions [28].

The GVM algorithm is a new classification algorithm proposed by Zhao in 2016 [3]. Since it contains the design risk minimization and Monte Carlo (MC) algorithm, it has strong generalization ability and has been successfully applied in phishing detection [29], Android malware detection [30], groundwater status forecasting [31], electricity demand prediction [32]. Yong *et al.* proposed a derivative-based Monte Carlo algorithm to accelerate the training of GVM based on the GVM [33]. Yong *et al.* applied the Monte Carlo neural network (MCNN) based on the GVM to electricity load forecast. Deep MCNNs with one, two and three hidden layers were designed at the same time. The results demonstrated that deeper MCNN performed better than shallow MCNN [34]. Yang *et al.* introduced a GVM approximate calculation

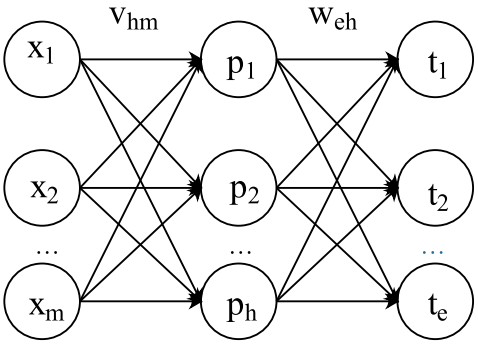


FIGURE 1. The structure of GVM.

system based on field programmable gate array, which can effectively accelerate the calculation of GVM [35].

III. OUR PROPOSED ALGORITHM

ALO is a new metaheuristic optimization algorithm proposed by Mirjalili in 2015 [27]. The high applicability of the ALO algorithm can be obtained from the experimental results of multi-modal and composite test functions in literature [27], which is also attributed to the random walk and roulette selection mechanism. The BALO algorithm is a binary form of the ALO algorithm. The position of the antlion is converted to 0 or 1, which is no longer a continuous value. Also, the initial position is constructed by a random method. Because of its novelty and underlying excellent characteristics, the proposed algorithm CFGVM combines cost-sensitive learning method and feature selection method based on BALO and GVM. Firstly, The BALO algorithm is used to improve the GVM algorithm to a cost-sensitive CGVM algorithm, and then the BALO algorithm is used to search the optimal feature subset to further improve the classification performance, as described in subsections 3.1 and 3.2.

A. CGVM

$$k_h = \sum_{m=1}^{M} v_{hm} \cdot x_m + b_h, \quad h = 1, 2, \ldots, H \tag{1}$$

$$p_h = f_{xp}(\beta_h \cdot k_h), \quad h = 1, 2, \ldots, H \tag{2}$$

$$t_e = f_{pt}(\sum_{h=1}^{H} w_{eh} \cdot p_h), \quad e = 1, 2, \ldots, E \tag{3}$$

The structure of GVM is composed of three layers: input layer, hidden layer and output layer, as shown in Fig 1. p_h and t_e represent the outputs of the hidden layer and the output layer, as deducted by Formula 2 and 3, respectively. x_m $(m = 1, 2, \ldots M)$ denotes the input layer units. $v_{hm}[h][m]$ is the weight connecting the h-th hidden node and the m-th input node. $w_{eh}[e][h]$ represents the weight connecting the h-th hidden layer and the e-th output layer. b_h $(h = 1, 2, \ldots, H)$ stands for the bias of the hidden layer. β_h is the transfer function coefficient. $f_{xp}()$ denotes the activation function of the hidden layer. $f_{pt}()$ denotes the activation function of the output layer. $f_{pt}()$ is set as a linear activation function. v_{hm} is initialized as decimals between -1 and $+1$. For the sake of simplicity, w_{eh} is set to be a random value $+1$ or -1 in the initialization. After that, we will not change w_{eh} and $f_{pt}()$. In the training phase, we only need to adjust the weight matrices v_{hm}, b_h and β_h. In the GVM algorithm, we adopt the Monte Carlo training algorithm. In other words, we only randomly change one weight of one parameter matrix (v_{hm}, b_h or β_h) in a relatively small range every time, and GVM will accept the weight change while the overall cost is reduced. In this case, the output is insensitive to the small change of the input. On the other hand, the GVM algorithm introduces the design risk to improve the generalization ability. The second derivative can be used to express the smoothness of the model. GVM algorithm uses the second derivative to control the smoothness of the model [3]. Considering the response of the h node in the hidden layer to the change of the l-th and m-th dimension in the input layer, the corresponding second derivative is:

$$\frac{\partial^2 p_h}{\partial x_l \partial x_m} = \beta_h^2 v_{hl} v_{hm} f_h''(\beta_h k_h) \tag{4}$$

the corresponding second derivative of the e-th dimension in the output layer to the change of the l-th and m-th dimension in the input layer,

$$\frac{\partial^2 t_e}{\partial x_l \partial x_m} = \sum_{h=1}^{H} w_{eh} \frac{\partial^2 p_h}{\partial x_l \partial x_m} \tag{5}$$

The smoothness (SR) of the whole network is a linear superposition of the smoothness of all nodes.

$$SR = \left\langle \sum_{h=1}^{H} w_{eh} \frac{\partial^2 p_h}{\partial x_l \partial x_m} \right\rangle \tag{6}$$

Hence, we can control the smoothness of the network by controlling the amplitude of $\beta_h, b_h, v_{hm}, f_h''$. b_h, v_{hm}, f_h'' are all limited in a small range in the initialization phase. We only need to adjust the range of β_h. In this way, we can control the smoothness of the whole network to reduce the design risk by controlling the range of β_h. β_h is the control parameter in the GVM algorithm. In the initialization phase, v_{hm} is set to be a random decimal between $+1$ and -1. b_h is set to be a random decimal between $+1$ and -1. $f_h()$ is set to be *tanh*. Pragmatically, the number of hidden layer nodes is set to 10 times the number of samples [3]. In the classification problem of the original GVM algorithm, the penalty factor of the loss function is the same for the majority class and minority class. The loss function formula is as follows:

$$\Delta F = \frac{1}{Q \cdot E} \cdot \sum_{q=1}^{Q} \sum_{e=1, t_e \cdot t_{e0} < g}^{E} (t_e \cdot t_{e0} - g)^2 \tag{7}$$

E represents the number of neurons in the output layer. g denotes the separating margin, which mainly separates the two types of samples, somewhat like the hyperplane of SVM. In the classification problem, we mainly control the output in

the output layer by adjusting the value of g. Therefore, g is also the control parameter in the GVM algorithm.

For the two-class problem, E is set as 2. Q denotes the number of samples. The t_e represents the actual output of the neural network. The t_{e0} represents the label of the neural network. Suppose there are two categories: Category 1 and Category 2. If the true value of the category is 1, then t_{10} and t_{20} are set to $\{1, -1\}$, otherwise $\{-1, 1\}$. More details of GVM can refer to [3].

ALO is a new metaheuristic optimization algorithm that simulates the biological behavior of antlions hunting ants. When hunting, the antlion digs a trap in the sand and hides at the bottom of the cave to wait for its prey. Once the ant enters the trap, to prevent it from escaping, the antlion will immediately dig out the sand and slide it into the bottom of the cave to hunt. The above process can be summarized into five steps: the movement of ants, the construction of traps, the trapping of ants in traps, the capture of prey, and the reconstruction of traps. The standard ALO algorithm follows the following principles: (1) Each ant searches around the selected antlion by random walk; (2) The range boundary of random walk is affected by the shrinkage of the antlion trap; (3) The antlion builds pits of different sizes according to their adaptability, and more ants enter the trap in larger pits; (4) Each ant is affected by both the selected antlion and the elite antlion in each iteration; (5) If the ant is more adaptable than the selected antlion, it means that it is captured by the antlion; (6) The antlion repositioned the captured prey and built pits to increase the probability of capturing another prey.

The process of ALO algorithm is as follows:

(1) To determine the initial population of ants and antlions. m is population number, dim represents variable dimension, t is current iteration number, $Maxiter$ stands for maximum iteration number, $antposition$ is the ant position, $antlionposition$ is the antlion position, $Elite$ is the position of the elite antlion, $Antlionfitness$ is the antlion's fitness value, $Elitefitness$ is the fitness value of the elite antlion;

(2) The positions of ants and antlions are randomly initialized within the boundary of the solution space, and the fitness values of all the populations are calculated according to the objective function sorted in descending order. We define the antlion with the best fitness value in the antlion population as the elite antlion;

(3) Each ant is randomly matched with an antlion by roulette. The upper and lower bounds c^t, d^t, c_i^t, d_i^t of the range are updated according to the matched position Formula 10 and 11. The ants are randomly walked around the selected antlion and the elite antlion according to Formula 8, 12 and 13. Then the average value is obtained by Formula 14 and used as the initial position of the next generation of the ant.

$$C(antposition^d) = [0, cumsum(2r(t_1) - 1), \ldots, cumsum(2r(Maxiter) - 1)] \quad (8)$$

$$r(t) = \begin{cases} 1 & if\ rand > 0.5 \\ 0 & otherwise \end{cases} \quad (9)$$

$$c_i^t = antlionposition_j^t + c^t \quad (10)$$

$$d_i^t = antlionposition_j^t + d^t \quad (11)$$

$$c^t = \frac{c^t}{I} \quad (12)$$

$$d^t = \frac{d^t}{I} \quad (13)$$

$$antposition^d(t) = \frac{R_A^t + R_E^t}{2} \quad (14)$$

The ant's random walk around the antlion can be regarded as the search process for feasible regions by search agents by Formula 8. $C(antposition^d)$ represents the set of steps of ant random walk, $cumsum$ is the cumulative sum of ant's wandering position, t is the number of iterations. $Maxiter$ is the maximum number of iterations. $r(t)$ is a custom random function, $rand$ is a random number uniformly distributed on [0,1]. The position of the antlion affects the boundary range of the area where the ant travels by Formula 10 and 11. c^t represents the minimum of all variables at t-th iteration, d^t denotes the vector including the maximum of all variables at t-th iteration, c_i^t represents the minimum of all variables for i-th ant, d_i^t is the maximum of all variables for i-th ant, I is a constant that increases with the number of iterations. When an ant falls into a trap, the antlion will raise sand, sharply reducing the range of the ant's movement to prevent the ant from escaping, that is, with the increase of iteration times, the upper and lower bounds decrease by Formula 12 and 13. $antposition^d(t)$ denotes the position of the i-th ant at the (t+1)-th iteration. R_A^t is the random walk around the antlion selected by the roulette wheel at t-th iteration, R_E^t represents the random walk around the elite at t-th iteration.

(4) For each iteration, the initial values of ant and antlion populations are reassigned and their fitness values are recalculated. The position of the best fitness value is taken as the position of the new generation of elite antlion.

(5) Determine whether the maximum number of iterations is reached, if it is reached, the iteration ends, and the final result is output; if each maximum number of iterations is not reached, repeat step (3).

More details of ALO can refer to [27].

$$x^d = \begin{cases} x_1^d & if\ (rand) \geq 0.5 \\ x_2^d & otherwise \end{cases} \quad (15)$$

$$x_{out}^d = \begin{cases} x_{in}^d & if\ (rand1) \geq r \\ rand2 & otherwise \end{cases} \quad (16)$$

$$r = 0.9 + \frac{-0.9 * (i-1)}{IterMax - 1} \quad (17)$$

BALO is a binary variant of the ALO. In the BALO algorithm, each dimension of the search space is limited to 0 or 1. In the ALO algorithm, each ant updates its position by averaging the selected antlion by the roulette wheel and the elite antlion. While the average operator is replaced by the crossover operation in the BALO algorithm. The crossover operation here refers to simple random crossover, switching between two input vectors with different probabilities by

Formula 15. $CW1$ represents the attraction of an ant by the elite antlion. It can also stochastic mutation around a selected antlion with a suitable mutation rate by Formula 16. $CW2$ represents the attraction of an ant by the selected antlion by the roulette wheel selection method. x_{out}^{d} is the d dimension value for the output vector from mutation, x_{in}^{d} is the input vector to be mutated, $rand1$ and $rand2$ are two random numbers in the range [0,1]. r is the mutation rate at the i-th iteration.

Here we propose a cost-sensitive algorithm CGVM by setting different cost weight-based loss functions based on GVM. The new loss function formula is as follows:

$$\Delta F = \frac{1}{Q \cdot E} \cdot \sum_{q=1}^{Q} \sum_{e=1, t_e \cdot t_{e0} < g}^{E} \times C_{ij}(t_e \cdot t_{e0} - g)^2 + C_{ji}(t_e \cdot t_{e0} - g)^2 \quad (18)$$

It is assumed here that i represents the minority class, j represents the majority class, then C_{ij} represents the cost weight of misclassification of minority class samples, and the cost weight of misclassification of majority class samples is represented by C_{ji}. Since we assume that the value of C_{ji} is 1, it is clear that the value of C_{ij} should be greater than 1, and we use the BALO algorithm to choose the optimal C_{ij} value. In the CGVM algorithm, the position of the antlion in the BALO algorithm represents the value of C_{ij}. That is, when using BALO to select the cost weights of the minority class, the position of antlion represents the cost weights to be optimized. The number of integer bits is set to 4 digits, and the decimal digit is set to 7 digits. The CGVM is described in Algorithm 1. In the selection of the optimal value, we adopt $(1-G\text{-}mean)$ as the fitness function. The comparison algorithm also uses $(1-Accuracy)$ and $(1-F\text{-}mean)$ as the fitness function, which we will discuss in detail in Section III.C.

B. SELECT THE OPTIMAL FEATURE SUBSET

We can get the optimal value using the CGVM algorithm. After that, we obtain the optimal feature by adopting the BALO as the feature selection algorithm. Specifically, the position of the antlion in the BALO algorithm represents the number of features in Algorithm 2. When the value of the antlion position is 1, it means that the feature is selected; but, when the value of the antlion position is 0, it means that the feature is not selected. The specific process is described in Algorithm 2.

C. EVALUATION CRITERIA AND FUNCTION

As in [36]–[39], *Accuracy*, *True-positive rate* (*TPR*), *False-positive rate* (*FPR*), *AUC*, *F-measure*, *G-mean* are the most commonly used evaluation metrics. We also adopt these evaluation metrics to evaluate the performance of different algorithms, as shown in equations 19, 20, 21, 24 and 25, respectively. In this paper, the minority class represents the positive class, the majority class represents the negative class. *Accuracy* is the proportion of all samples correctly classified. *True Positive* (*TP*) and *True negative* (*TN*) are the correctly classified samples belonging to the positive and the negative class respectively. Whereas, *False positive* (*FP*) and *False negative* (*FN*) is the mistakenly classified samples belonging to the negative and the positive class respectively. *TPR* represents the value of predicted positive classified correctly. *FPR* represents the value negative class samples mistakenly

Algorithm 1 CGVM

Input: m_1 number of ants, m_2 number of antlions, dim dimension of antlions, dim dimension of ants, t number of iterations, $Maxiter$ maximum number of iterations, $fitness$ fitness function

Output: C_{best} best cost weight of minority class

1: Dividing training sets and test sets by stratified random sampling
2: Randomly initialize the position of the antlions and ants on the training set
3: **if** $random() \geq 0.5$ **then**
4: $\quad antposition_i = 1$
5: **else**
6: $\quad antposition_i = 0, i = 1 \ldots dim$
7: **end if**
8: **if** $random() \geq 0.5$ **then**
9: $\quad antlionposition_i = 1$
10: **else**
11: $\quad antlionposition_i = 0, i = 1 \ldots dim$
12: **end if**
13: Using the formula 18 as the loss function, $(1-G\text{-}mean)$ as the fitness, and the GVM algorithm as the classifier, calculate the fitness values of all the antlions and ants.
14: Sort all the antlions to get the best antlion position
15: **while** $t < Maxiter$ **do**
16: $\quad$ Calculate the mutation rate t by Formula 17
17: $\quad$ **for** each ant **do**
18: $\quad\quad$ Choose an antlion by Roulette
19: $\quad\quad$ Mutation operation of the selected antlion by Formula 16 becomes $CW1$
20: $\quad\quad$ Mutant operation of the ant lion with the best fitness value by Formula 16 becomes $CW2$
21: $\quad\quad$ Cross the operation of $CW1$ and $CW2$ by Formula 15 to get the position of the new ant
22: $\quad$ **end for**
23: $\quad$ Calculate the fitness value of all ants
24: $\quad$ If the fitness value of the corresponding ant is better than the fitness value of the antlion, the position of the antlion is replaced with the current ant.
25: $\quad$ If the fitness value of the antlion is better than the fitness value of the current elite, adjust the elite to the current antlion
26: **end while**
27: Get the best antlion' position p_{best} and its fitness value f_{best}
28: Convert to the best minority class cost C_{best} based on the best antlion' position p_{best}

Algorithm 2 Feature Selection Using the BALO Method Based on the Algorithm CGVM (CFGVM)

Input: m_1 number of ants, m_2 number of antlions, *dim* dimension of antlions, *dim* dimension of ants, t number of iterations, *Maxiter* maximum number of iterations, *fitness* fitness function

Output: *featuresub* Optimal feature subset

Randomly initialize the position of all antlions and ants for the training set divided by CGVM algorithm
if $random() \geq 0.5$ **then**
    $antposition_i = 1$
else
    $antposition_i = 0, i = 1 \ldots dim$
end if
if $random() \geq 0.5$ **then**
    $antlionposition_i = 1$
else
    $antlionposition_i = 0, i = 1 \ldots dim$
end if
The current feature subset *featuresub* is obtained according to the position of the antlion, and a new training set is obtained by feature selection of the training set.
Formula 18 as loss function, fitness 3 as fitness and CGVM algorithm as classifier to calculate the fitness values of all antlions and ants.
Sort the fitness values of all antlions to get the best antlion' position
while $t <$ *Maxiter* **do**
    Calculate the mutation rate *mt* by Formula 17
    for each ant **do**
        Choose an antlion by Roulette
        Mutation operation of the selected antlion by Formula 16 becomes $CW1$
        Mutant operation of the ant lion with the best fitness value by Formula 16 becomes $CW2$
        Cross the operation of $CW1$ and $CW2$ by Formula 15 to get the position of the new ant
    end for
    Calculate the fitness value of all ants
    If the fitness value of the corresponding ant is better than the fitness value of the antlion, the position of the antlion is replaced with the current ant.
    If the fitness value of the antlion is better than the fitness value of the current elite, adjust the elite to the current antlion
end while
Get the best antlion' position p_{best} and its fitness value f_{best}
Convert to the best feature subset *featuresub* based on the best antlion' position p_{best}

predicted as positive. *Precision* is the ratio of *TP* to all the positive results, *Recall* is the proportion of *TP* in all the positive class samples. *F-measure* is the weighted harmonic mean of *Precision* and *Recall*. *G-mean* is also a comprehensive indicator. The closer to 1, the better. *AUC* stands for the area under the ROC curve, the bigger, the better. *fitness*3 is used as the fitness function in the feature selection process in this paper, as shown in Formula 28. At the same time, *fitness*1 and *fitness*2 are also used as the fitness function in the comparative algorithms, as shown in Formula 26 and 27, respectively. The len (featuresub) represents the number of selected features, the len (feature) represents the total number of features, and the α parameter is used to adjust the weight relationship between the number of features and the error rate. In this paper, α is set to 0.01, the same defined in [4].

TABLE 2. Data descriptions used in the experiment.

Name	Minority	Majority	Nf	Ss	MiSs	MaSs	IR	Source
breast_tissue	"car" and "fad"	All other	9	106	36	70	1.94	UCI
bupa	"1"	"2"	6	345	145	200	1.38	UCI
cleveland	positive	negative	12	303	35	268	7.66	Keel
ecoli01VS235	positive	negative	7	244	24	220	9.17	Keel
glass4	containers	ALL other	9	244	13	201	15.47	UCI
Wisconsin	Malignant	Benign	9	683	239	444	1.86	UCI
glass6	headlamps	ALL other	9	244	29	185	6.38	Keel
glass016vs5	positive	negative	9	184	9	175	19.44	Keel
newthyroid1	"1"	"2"	5	215	35	180	5.14	Keel
shuttlec2vsc4	"1"	"2"	9	129	6	123	20.5	Keel
ecoli034vs5	"1"	"2"	7	200	20	180	9	Keel

Nf represents the number of feature; Ss stands for the sample size; MiSs is the minority sample size; MaSs represents the majority sample size; IR stands for the imbalance rate.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{19}$$

$$TPR = \frac{TP}{TP + FN} \tag{20}$$

$$FPR = \frac{FP}{TN + FP} \tag{21}$$

$$Precision = \frac{TP}{TP + FP} \tag{22}$$

$$Recall = TPR = \frac{TP}{TP + FN} \tag{23}$$

$$F\text{-}measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{24}$$

$$G\text{-}mean = \sqrt{\frac{TP}{TP + TN} * \frac{TN}{TN + FP}} \tag{25}$$

$$fitness1 = (1 - \alpha)(1 - Accuracy) + \alpha(len(featuresub)/len(feature)) \tag{26}$$

$$fitness2 = (1 - \alpha)(1 - F\text{-}measure) + \alpha(len(featuresub)/len(feature)) \tag{27}$$

$$fitness3 = (1 - \alpha)(1 - G\text{-}mean) + \alpha(len(featuresub)/len(feature)) \tag{28}$$

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. DATASETS AND EXPERIMENTAL ENVIRONMENT

To evaluate the CFGVM algorithm, we conducted experiments on eleven datasets, where four of them were from UCI repository [40] (*bupa*, *breast_tissue*, *glass*4,

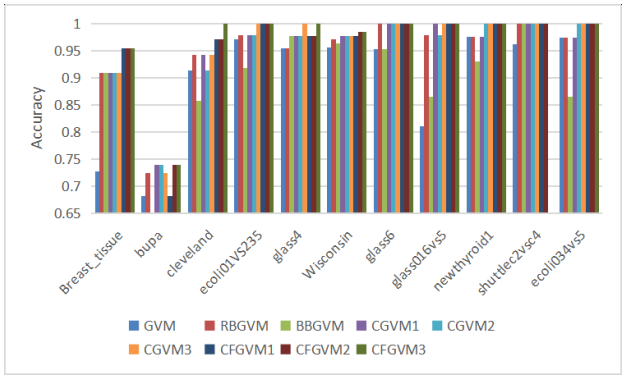


FIGURE 2. Display of *Accuracy* value for nine algorithms on eleven datasets.

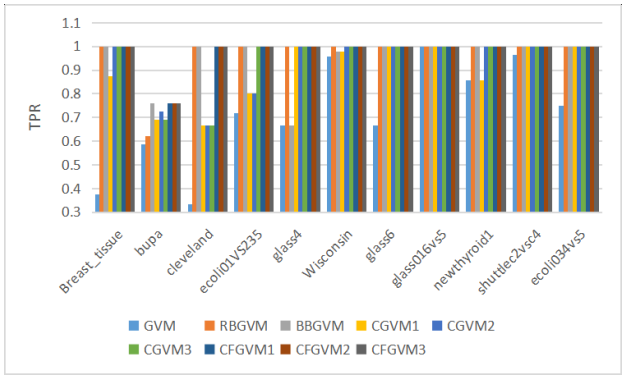


FIGURE 3. Display of *TPR* value for nine algorithms on eleven datasets.

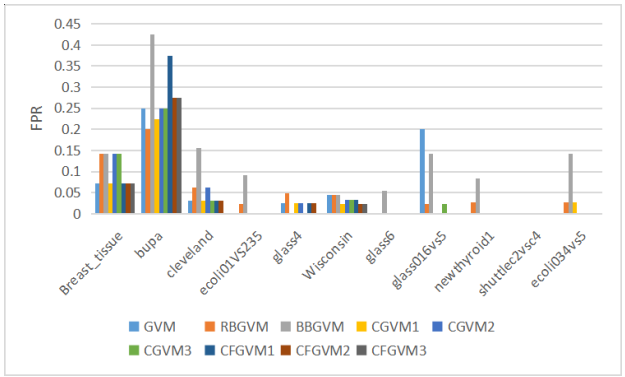


FIGURE 4. Display of *FPR* value for nine algorithms on eleven datasets.

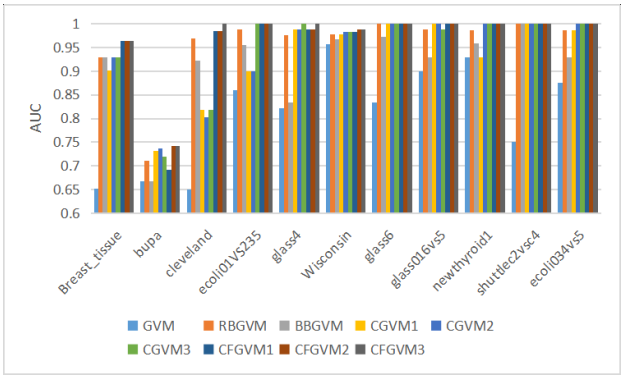


FIGURE 5. Display of *AUC* value for nine algorithms on eleven datasets.

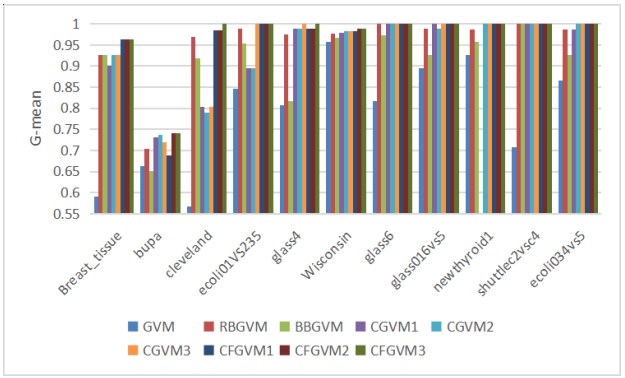


FIGURE 6. Display of *G-mean* value for nine algorithms on eleven datasets.

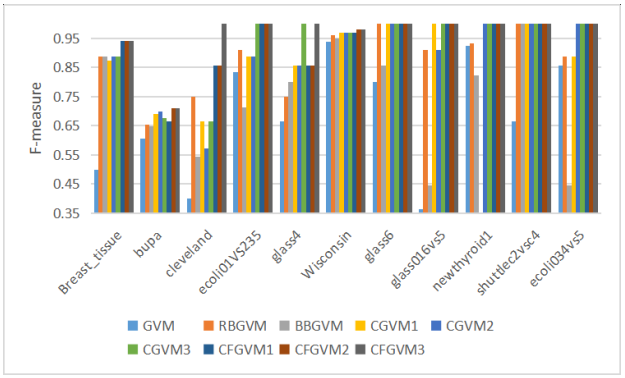


FIGURE 7. Display of *F-measure* value for nine algorithms on eleven datasets.

and *Wisconsin* datasets), and seven from Keel [41] (*cleveland*, *ecoli*01*VS*235, *glass*6, *glass*016*vs*5, *newthyroid*, *shuttlec*2*vsc*4, and *ecoli*034*vs*5 datasets). The experimental environment is a PC configured with Intel Core i5 7500 3.4 GHz CPU, 8GB SmartCache, and MS Windows10 64-bit OS. All algorithms are executed in the same experimental environment, and the final results are the averages of 20 executions. Experiments are implemented using Matlab programming language in Matlab R2017a environment. Details of the datasets are shown in Table 2.

B. COMPARISON OF CFGVM ALGORITHM AND ITS SIMILAR ALGORITHM

To verify the performance and effectiveness of the proposed algorithm, comparison to similar algorithms are shown in Table 3. In all the oversampling algorithms, we stop sampling as the number of minority class samples is the same as the majority class samples by the oversampling algorithm. The complete dataset is split as: 80% for training and 20% for testing by random stratified sampling. The parameters of the BALO algorithm and the GVM algorithm are the same in different comparison algorithms in the same dataset. In the selection of the optimal weight using BALO, the number of iterations, and the scale of the population is set to 10. In the selection of the significant feature by BALO, the number of iterations, and the scale of the population is set to 15. In the BBMCA algorithm, the number of iterations and the scale of the population is set to 10 in the selection of the optimal weight using BPSO. In the selection of the significant

TABLE 3. Description of contrast algorithms.

Algorithm	Detailed description
GVM	GVM classification algorithm
RBGVM	The imbalance rate is used as the cost weight of minority classes, BALO is used as feature selection algorithm and GVM is used as a classification algorithm, the fitness function of the selected feature is the formula 28
BBGVM	BPSO is used to improve the GVM algorithm to obtain the cost weights of minority classes, then BPSO is used as the feature selection algorithm to select the features, GVM is used as the classification algorithm. The fitness function of the cost of selection is $(1-G\text{-}mean)$, and the fitness function of feature selection is the formula 28
CGVM1	BALO is used to improve the GVM algorithm to obtain the cost weight of minority classes. $(1 - Accuracy)$ is adopted as the formula of fitness function
CGVM2	BALO is used to improve the GVM algorithm to obtain the cost weight of minority classes. $(1-F\text{-}measure)$ is adopted as the formula of fitness function
CGVM3	BALO is used to improve the GVM algorithm to obtain the cost weight of minority classes. $(1-G\text{-}mean)$ is adopted as the formula of fitness function
CFGVM1	BALO is used to improve the GVM algorithm to obtain the cost weight of minority classes, Then BALO is used as feature selection algorithm to select features. GVM is used as classification algorithm. $(1 - Accuracy)$ is adopted as the fitness function of the cost of selection, The formula 26 is adopted as the fitness function of feature selection
CFGVM2	BALO is used to improve the GVM algorithm to obtain the cost weight of minority classes, Then BALO is used as a feature selection algorithm to select features. GVM is used as classification algorithm. $(1-F\text{-}measure)$ is adopted as the fitness of the cost of selection, The formula 27 is adopted as the fitness function of feature selection
CFGVM3	BALO is used to improve the GVM algorithm to obtain the cost weight of minority classes, Then BALO is used as a feature selection algorithm to select features. GVM is used as classification algorithm. $(1-G\text{-}mean)$ is adopted as the fitness function of the cost of selection, The formula 28 is adopted as the fitness function of feature selection

feature by BPSO, the number of iterations and the scale of the population is set to 15, c1 and c2 are both set to 1.49445, the maximum speed and minimum speed are both set to 1. β_h is set to 1.

The comparison of experimental results between the proposed algorithm CFGVM and similar algorithms are shown in Table 4 and 5. We can observe that the CFGVM algorithm outperforms other similar algorithms in the classification performance of imbalanced datasets. Specific analysis is discussed below:

(1) The performance of the CFGVM1, CFGVM2 and CFGVM3 algorithm are the same in dataset *glass*6, *ecoli*01*VS*235, *Breast_tissue*, *glass*016*vs*5, *newthyroid*1, *shuttlec*2*vsc*4 and *ecoli*034*vs*5. In particular, the classification indicators *Accuracy*, *TPR*, *FPR*, *AUC*, *G-mean*, *F-measure* of the three algorithms are all 1, 1, 0, 1, 1, 1 in dataset *ecoli*01*VS*235, *glass*6, *glass*016*vs*5, *newthyroid*1, *shuttlec*2*vsc*4 and *ecoli*034*vs*5. Six evaluation indicators of CFGVM3 algorithm are better than those of CFGVM1 and CFGVM2 on all datasets, indicating that the *G-mean* can be used to obtain better classification performance than the *Accuracy* and *F-measure* in the fitness function.

(2) It can be seen from Tables 4 and 5 that the performances of CFGVM series (CFGVM1, CFGVM2, CFGVM3) have improved compared with CGVM series (CGVM1, CGVM2, CGVM3) on ten datasets, except CGVM1 and CFGVM1 on the *bupa* data set. It shows that the feature selection algorithm is helpful to improve the classification performance of imbalanced classification.

(3) Comparing the results of the RBGVM and CFGVM3, except that *Accuracy*, *TPR*, *FPR*, *AUC*, *G-Mean* and *F-measure* are all 1, 1, 0, 1, 1, 1 on *glass*6, *shuttlec*2*vsc*4 dataset, the *Accuracy*, *TPR*, *FPR*, *AUC*, *G-mean* and *F-measure* values of CFGVM3 are all better than those of RBGVM on other nine data sets, which shows that CFGVM3 is better than using imbalanced rate as cost weight in the classification performance of imbalanced datasets.

(4) Based on the comparison between the BBGVM and CFGVM3, except that the *TPR* value in dataset *Breast_tissue*, *bupa*, *cleveland*, *ecoli*01*VS*235, *glass*, *glass*016*vs*5, *newthyroid*, *shuttlec*2*vsc*4 and *ecoli*034*vs*5 are the same, other five evaluations of CFGVM3 are superior to BBGVM on all data sets, which shows that BALO algorithm has better classification performance than BPSO algorithm in choosing the optimal weight and feature.

(5) Compared with all other algorithms, we can see that the performances of other algorithms are significantly improved compared with that of single classification algorithm GVM. It demonstrates that feature selection and cost-sensitive learning algorithms can effectively improve the classification performance of imbalanced classification.

Fig.2-Fig.7 have shown the *Accuracy*, *TPR*, *FPR*, *AUC*, *G-mean* and *F-measure* for nine algorithms on eleven testing datasets, respectively. It can be seen that the proposed CFGVM3 obtains the best *Accuracy*, *TPR*, *AUC*, *G-mean*, *F-measure* results among other algorithms on all datasets. From Fig.4 we can observe that the CFGVM3 achieves the lowest *FPR* in 10 out of 11 datasets.

The above experiment shows that the result of CFGVM3 is better than that of CFGVM1 and CFGVM2, so this subsection only compares CFGVM3 with other algorithms. Tables 6, 7, 8 and 9 show the comparisons between the CFGVM3 and

TABLE 4. Experimental results of CFGVM and similar algorithms on the test data sets.

Dataset	Method	Accuracy	TPR	FPR	AUC	G-mean	F-measure
Breast_tissue	GVM	0.7273	0.375	**0.0714**	0.6518	0.59	0.5
	RBGVM	0.9091	**1**	0.1429	0.9286	0.9258	0.8889
	BBGVM	0.9091	**1**	0.1429	0.9286	0.9258	0.8889
	CGVM1	0.9091	0.875	**0.0714**	0.9018	0.9014	0.875
	CGVM2	0.9091	**1**	0.1429	0.9286	0.9258	0.8889
	CGVM3	0.9091	**1**	0.1429	0.9286	0.9258	0.8889
	CFGVM1	**0.9545**	**1**	**0.0714**	**0.9643**	**0.9636**	**0.9412**
	CFGVM2	**0.9545**	**1**	**0.0714**	**0.9643**	**0.9636**	**0.9412**
	CFGVM3	**0.9545**	**1**	**0.0714**	**0.9643**	**0.9636**	**0.9412**
bupa	GVM	0.6812	0.5862	0.25	0.6681	0.6631	0.6071
	RBGVM	0.7246	0.6207	**0.2**	0.7103	0.7047	0.6545
	BBGVM	0.6522	**0.7586**	0.425	0.6668	0.6605	0.6471
	CGVM1	**0.7391**	0.6897	0.225	0.7323	0.7311	0.6897
	CGVM2	**0.7391**	0.7241	0.25	0.7371	0.737	0.7
	CGVM3	0.7246	0.6897	0.25	0.7198	0.7192	0.678
	CFGVM1	0.6812	**0.7586**	0.375	0.6918	0.6886	0.6667
	CFGVM2	**0.7391**	**0.7586**	0.275	**0.7418**	**0.7416**	**0.7097**
	CFGVM3	**0.7391**	**0.7586**	0.275	**0.7418**	**0.7416**	**0.7097**
cleveland	GVM	0.9143	0.3333	0.0313	0.651	0.5683	0.4
	RBGVM	0.9429	**1**	0.0625	0.9688	0.9682	0.75
	BBGVM	0.8571	**1**	0.1563	0.9219	0.9186	0.5455
	CGVM1	0.9429	0.6667	0.0313	0.8177	0.8036	0.6667
	CGVM2	0.9143	0.6667	0.0625	0.8021	0.7906	0.5714
	CGVM3	0.9429	0.6667	0.0313	0.8177	0.8036	0.6667
	CFGVM1	0.9714	**1**	0.0313	0.9844	0.9843	0.8571
	CFGVM2	0.9714	**1**	0.0313	0.9844	0.9843	0.8571
	CFGVM3	**1**	**1**	**0**	**1**	**1**	**1**
ecoli01VS235	GVM	0.9714	0.72	**0**	0.86	0.8465	0.8333
	RBGVM	0.9796	**1**	0.0227	0.9886	0.9886	0.9091
	BBGVM	0.9184	**1**	0.0909	0.9545	0.9535	0.7143
	CGVM1	0.9796	0.8	**0**	0.9	0.8944	0.8889
	CGVM2	0.9796	0.8	**0**	0.9	0.8944	0.8889
	CGVM3	**1**	**1**	**0**	**1**	**1**	**1**
	CFMCA1	**1**	**1**	**0**	**1**	**1**	**1**
	CFMCA2	**1**	**1**	**0**	**1**	**1**	**1**
	CFMCA3	**1**	**1**	**0**	**1**	**1**	**1**
glass4	GVM	0.9545	0.667	0.0244	0.8211	0.8065	0.6667
	RBGVM	0.9545	**1**	0.0488	0.9756	0.9753	0.75
	BBGVM	0.9773	0.667	**0**	0.8333	0.8165	0.8
	CGVM1	0.9773	**1**	0.0244	0.9878	0.9877	0.8571
	CGVM2	0.9773	**1**	0.0244	0.9878	0.9877	0.8571
	CGVM3	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM1	0.9773	**1**	0.0244	0.9878	0.9877	0.8571
	CFGVM2	0.9773	**1**	0.0244	0.9878	0.9877	0.8571
	CFGVM3	**1**	**1**	**0**	**1**	**1**	**1**
Wisconsin	GVM	0.9562	0.9583	0.0449	0.9567	0.9567	0.9388
	RBGVM	0.9708	**1**	0.0449	0.9775	0.9773	0.96
	BBGVM	0.9635	0.9792	0.0449	0.9671	0.967	0.9495
	CGVM1	0.9781	0.9792	**0.0225**	0.9783	0.9783	0.9691
	CGVM2	0.9781	**1**	0.0337	0.9831	0.983	0.9697
	CGVM3	0.9781	**1**	0.0337	0.9831	0.983	0.9697
	CFGVM1	0.9781	**1**	0.0337	0.9831	0.983	0.9697
	CFGVM2	**0.9854**	**1**	**0.0225**	**0.9888**	**0.9887**	**0.9796**
	CFGVM3	**0.9854**	**1**	**0.0225**	**0.9888**	**0.9887**	**0.9796**
glass6	GVM	0.9535	0.6667	**0**	0.8333	0.8165	0.8
	RBGVM	**1**	**1**	**0**	**1**	**1**	**1**
	BBGVM	0.9535	**1**	0.0541	0.973	0.9726	0.8571
	CGVM1	**1**	**1**	**0**	**1**	**1**	**1**
	CGVM2	**1**	**1**	**0**	**1**	**1**	**1**
	CGVM3	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM1	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM2	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM3	**1**	**1**	**0**	**1**	**1**	**1**

previous algorithms, mainly including A-SUWO [42], IMC-Stacking [43], SMOTE-IPF [44], FRB+CHC+SVM [45], LCMine+CAEP [46], CO^2RBFN-LMS, CO^2RBFN-SVD [47]. ''−'' means that information is unavailable. The data sets are divided in the same way as those in the comparative literature. Except for that, the dataset *Breast_tissue* is a 4-fold

TABLE 5. Experimental results of CFGVM and similar algorithms on the test data sets.

Dataset	Method	Accuracy	TPR	FPR	AUC	G-mean	F-measure
glass016vs5	GVM	0.8108	**1**	0.2	0.9	0.8944	0.3636
	RBGVM	0.9796	**1**	0.0227	0.9886	0.9886	0.9091
	BBGVM	0.8649	**1**	0.1429	0.9286	0.9258	0.4444
	CGVM1	**1**	**1**	**0**	**1**	**1**	**1**
	CGVM2	0.9796	**1**	0.0227	0.9886	0.9886	0.9091
	CGVM3	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM1	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM2	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM3	**1**	**1**	**0**	**1**	**1**	**1**
newthyroid1	GVM	0.9767	0.8571	**0**	0.9286	0.9258	0.9231
	RBGVM	0.9767	**1**	0.0278	0.9861	0.986	0.9333
	BBGVM	0.9302	**1**	0.0833	0.9583	0.9574	0.8235
	CGVM1	0.9767	0.8571	**0**	0.9286	0.9258	0.9231
	CGVM2	**1**	**1**	**0**	**1**	**1**	**1**
	CGVM3	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM1	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM2	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM3	**1**	**1**	**0**	**1**	**1**	**1**
shuttlec2vsc4	GVM	0.963	0.5	**0**	0.75	0.7071	0.6667
	RBGVM	**1**	**1**	**0**	**1**	**1**	**1**
	BBGVM	**1**	**1**	**0**	**1**	**1**	**1**
	CGVM1	**1**	**1**	**0**	**1**	**1**	**1**
	CGVM2	**1**	**1**	**0**	**1**	**1**	**1**
	CGVM3	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM1	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM2	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM3	**1**	**1**	**0**	**1**	**1**	**1**
ecoli034vs5	GVM	0.975	0.75	**0**	0.875	0.866	0.8571
	RBGVM	0.975	**1**	0.0278	0.9861	0.986	0.8889
	BBGVM	0.8649	**1**	0.1429	0.9286	0.9258	0.4444
	CGVM1	0.975	**1**	0.0278	0.9861	0.986	0.8889
	CGVM2	**1**	**1**	**0**	**1**	**1**	**1**
	CGVM3	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM1	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM2	**1**	**1**	**0**	**1**	**1**	**1**
	CFGVM3	**1**	**1**	**0**	**1**	**1**	**1**

cross-validation, the other ten datasets are all 5-fold cross-validation. The best results of each index are expressed in black bold. Detailed analysis is given as follows:

(1) In dataset *Breast_tissue*, the CFGVM3 is relatively better than A-SUWO+SVM, A-SUWO+KNN, A-SUWO+LR, A-SUWO+LDA [42], in terms of *AUC* improves 4.76%, 6.26%, 1.76%, and 1.72%, respectively, improves 15.89%, 12.79%, 8.29%, and 13.39% in terms of *G-mean*. improves 19.35%, 16.85%, 11.45%, 15.95% in terms of *F-measure*. Therefore, the CFGVM3 algorithm outperforms the A-SUWO algorithm in the performance of imbalanced classification.

(2) When considering the *bupa* and *cleveland* datasets, it is observed that the CFGVM3 outperforms all algorithms in [44]. In particular, compared with the SMOTE-IPF [44], the CFGVM3 has a performance improvement of more than 35% in dataset cleveland in terms of *AUC*, which shows that the overall performance of CFGVM3 is better than SMOTE-IPF.

(3) Through analyzing the results in the dataset *ecoli*01*VS*235, we can conclude that the CFGVM3 outperforms all algorithms in [43]. Comparing the optimal values of several indicators in [43], the result of CFGVM3 is 9.08% higher than that of FCStacking algorithm [43] in terms of *Accuracy*, CFGVM3 is 4% higher than that of RUSBagging-RF algorithm [43] in terms of *TPR*, and CFGVM3 is 4.09% lower than that of LCStacking algorithm [43] in terms of *FPR*.

(4) According to *glass*4 dataset, the CFGVM3 is better than the results of all the algorithms in literature [43] in terms of *TPR* and *FPR*, but only slightly lower than the FCSStacking in terms of *Accuracy*.

(5) Focusing on *Wisconsin* dataset, the CFGVM3 algorithm is better than all the algorithms in the literature [43] in terms of *AUC*, *G-mean* and *F-measure*, except for the index of *FPR* that is slightly higher than EasyEnsemble. The CFGVM3 is better than the proposed algorithm in literature [43] in terms of *TPR* and *Accuracy*.

(6) Regarding *glass*6 dataset, the CFGVM3 algorithm is better than all the algorithms in literature [43] in terms of *Accuracy*, *TPR*, and *FPR*.

(7) In dataset *glass*016*vs*5, *newthyroid*1 and *shuttlec*2*vsc*4, the classification indicators *Accuracy*, *TPR*, *FPR*, *AUC*, *G-mean*, *F-measure* of the CFGVM3 algorithms are all 1, 1, 0, 1, 1, 1. We find that the CFGVM3 algorithm is distinctly better than other comparison algorithms in the six classification indicators.

TABLE 6. Experimental results of CFGVM and other existing algorithms on the *Breast_tissue*, *bupa*, *cleveland* dataset.

Dataset	Method	Accuracy	TPR	FPR	AUC	G-mean	F-measure
Breast_tissue	Random+SVM	-	-	-	0.815	0.704	0.634
	SMOTE+SVM	-	-	-	0.829	0.722	0.654
	Borderline SMOTE+SVM	-	-	-	0.834	0.741	0.677
	safe-level SMOTE+SVM	-	-	-	0.833	0.729	0.663
	SBC+SVM	-	-	-	0.806	0.749	0.695
	Cluster SMOTE+SVM	-	-	-	0.834	0.718	0.679
	CBOS+SVM	-	-	-	0.86	0.734	0.664
	MWMOTE+SVM	-	-	-	0.849	0.739	0.672
	A-SUWO+SVM	-	-	-	0.86	0.748	0.685
	Random+KNN	-	-	-	0.845	0.752	0.682
	SMOTE+KNN	-	-	-	0.849	0.763	0.697
	Borderline SMOTE+KNN	-	-	-	0.851	0.763	0.7
	safe-level SMOTE+KNN	-	-	-	0.846	0.771	0.706
	SBC+KNN	-	-	-	0.825	0.734	0.689
	Cluster SMOTE+KNN	-	-	-	0.859	0.795	0.738
	CBOS+KNN	-	-	-	0.844	0.76	0.698
	MWMOTE+KNN	-	-	-	0.856	0.766	0.7
	A-SUWO+KNN	-	-	-	0.845	0.779	0.71
	Random+LR	-	-	-	0.88	0.792	0.724
	SMOTE+LR	-	-	-	0.882	0.796	0.733
	Borderline SMOTE+LR	-	-	-	0.896	0.77	0.696
	safe-level SMOTE+LR	-	-	-	0.883	0.803	0.738
	SBC+LR	-	-	-	0.854	0.763	0.697
	Cluster SMOTE+LR	-	-	-	0.88	0.821	0.759
	CBOS+LR	-	-	-	0.892	0.806	0.739
	MWMOTE+LR	-	-	-	0.885	0.794	0.725
	A-SUWO+LR	-	-	-	0.89	0.824	0.764
	Random+LDA	-	-	-	0.899	0.762	0.707
	SMOTE+LDA	-	-	-	0.897	0.754	0.696
	Borderline SMOTE+LDA	-	-	-	0.882	0.752	0.698
	safe-level SMOTE+LDA	-	-	-	0.891	0.765	0.706
	SBC+LDA	-	-	-	0.873	0.72	0.677
	Cluster SMOTE+LDA	-	-	-	0.887	0.76	0.704
	CBOS+LDA	-	-	-	0.887	0.763	0.703
	MWMOTE+LDA	-	-	-	0.892	0.769	0.719
	A-SUWO+LDA	-	-	-	0.897	0.773	0.719
	CFGVM3	0.9134	0.9166	0.0882	**0.9142**	**0.914**	**0.8815**
bupa	C4.5	-	-	-	0.644	-	-
	SMOTE+C4.5	-	-	-	0.6688	-	-
	SMOTE-ENN+C4.5	-	-	-	0.6146	-	-
	SMOTE-TL+C4.5	-	-	-	0.6018	-	-
	SL-SMOTE+C4.5	-	-	-	0.6684	-	-
	B1-SMOTE+C4.5	-	-	-	0.686	-	-
	B2-SMOTE+C4.5	-	-	-	0.6361	-	-
	SMOTE-IPF+C4.5	-	-	-	0.6753	-	-
	CFGVM3	0.7913	0.8	0.215	**0.7351**	0.7896	0.7619
cleveland	C4.5	-	-	-	0.5258	-	-
	SMOTE+C4.5	-	-	-	0.5485	-	-
	SMOTE-ENN+C4.5	-	-	-	0.5722	-	-
	SMOTE-TL+C4.5	-	-	-	0.6433	-	-
	SL-SMOTE+C4.5	-	-	-	0.6007	-	-
	B1-SMOTE+C4.5	-	-	-	0.5475	-	-
	B2-SMOTE+C4.5	-	-	-	0.5666	-	-
	SMOTE-IPF+C4.5	-	-	-	0.6282	-	-
	CFGVM3	0.9941	1	0.00625	**0.9968**	0.9968	0.96

(8) Through analyzing the results in the dataset *ecoli*034*VS*5, we can conclude that the CFGVM3 outperforms all algorithms in [46] and [45].

With the observations above, we drew the following conclusion: 1) The cost-sensitive CGVM1, CGVM2, CGVM3 algorithms outperform the single classification algorithm GVM; 2) Compare with CGVM1, CGVM2, CGVM3 and CFGVM1, CFGVM2, CFGVM3, it indicates that the feature selection algorithm BALO can further improve the classification performance based on the cost-sensitive algorithms CGVM1, CGVM2, CGVM3; 3) By comparing CFGVM1, CFGVM2, CFGVM3, it can be found that *G-mean* can better evaluate the classification performance of imbalanced classification compared to *F-measure* and *Accuracy*; 4) Comparing the results of the BBGVM and CFGVM3, it shows that BALO is better than BPSO used for feature selection and cost weight selection in the imbalance classification; 5) Comparing the results between RBGVM and CFGVM 3, it is obvious that the cost weight selected by BALO is better than the imbalance rate;

TABLE 7. Experimental results of CFGVM and other existing algorithms on the *ecoli*01*VS*235, *glass*4, *Wisconsin*, *glass*6, *glass*016*vs*5 dataset .

Dataset	Method	Accuracy	TPR	FPR	AUC	G-mean	F-measure
ecoli01VS235	IMCStacking	0.8873	0.82	0.0455	-	-	-
	FLCStacking	0.8873	0.82	0.0455	-	-	-
	FCStacking	0.905	0.86	0.5	-	-	-
	LCStacking	0.8845	0.81	0.0409	-	-	-
	CLR	0.8914	0.86	0.0773	-	-	-
	RUSBagging-RF	0.8645	0.91	0.1809	-	-	-
	RUSBoosting-RF	0.8736	0.82	0.0727	-	-	-
	EasyEnsemble	0.8959	0.9091	0.1173	-	-	-
	CFGVM3	**0.9958**	**0.95**	**0**	0.975	0.9732	0.9714
glass4	IMCStacking	0.9225	0.9	0.055	-	-	-
	FLCStacking	0.9925	0.9	0.055	-	-	-
	FCStacking	0.8775	0.8	0.045	-	-	-
	LCStacking	0.7776	0.6	0.0449	-	-	-
	CLR	0.9201	0.9	0.0598	-	-	-
	RUSBagging-RF	0.9027	0.9	0.0946	-	-	-
	RUSBoosting-RF	0.8744	0.8334	0.0847	-	-	-
	EasyEnsemble	0.8826	0.8902	0.1251	-	-	-
	CFGVM3	0.9857	**1**	**0.015**	0.9925	0.9924	0.8933
Wisconsin	IMCStacking	0.9717	0.9749	0.0315	-	-	-
	FLCStacking	0.9696	0.9707	0.0315	-	-	-
	FCStacking	0.9696	0.9707	0.0315	-	-	-
	LCStacking	0.9555	0.929	0.018	-	-	-
	CLR	0.9719	0.9708	0.027	-	-	-
	RUSBagging-RF	0.978	0.9917	0.0357	-	-	-
	RUSBoosting-RF	0.978	0.9874	0.0314	-	-	-
	EasyEnsemble	0.9748	0.9662	**0.0167**	-	-	-
	CFGVM3	**0.9837**	**1**	0.025	0.9875	0.9873	0.9775
glass6	IMCStacking	0.9419	0.9	0.0162	-	-	-
	FLCStacking	0.9419	0.9	0.0162	-	-	-
	FCStacking	0.9392	0.9	0.0216	-	-	-
	LCStacking	0.9225	0.8667	0.0216	-	-	-
	CLR	0.9203	0.9	0.0594	-	-	-
	RUSBagging-RF	0.9284	0.9333	0.0766	-	-	-
	RUSBoosting-RF	0.9477	0.9332	0.0377	-	-	-
	EasyEnsemble	0.9095	0.9459	0.127	-	-	-
	CFGVM3	**0.9904**	**1**	**0.01**	0.9945	0.9945	0.9636
glass016vs5	SVM	-	-	-	0.8443	-	0.665
	SMOTE+SVM	-	-	-	0.8856	-	0.5674
	ADASYN+SVM	-	-	-	0.9186	-	0.6592
	sTL+SVM	-	-	-	0.8791	-	0.5601
	sSafe+SVM	-	-	-	0.8853	-	0.5668
	sRST+SVM	-	-	-	0.9221	-	0.6551
	sCHC+SVM	-	-	-	0.8979	-	0.7548
	EUSCHC+SVM	-	-	-	0.8071	-	0.4688
	AHC+SVM	-	-	-	0.8943	-	0.7273
	FRB+CHC+SVM	-	-	-	0.9186	-	0.7692
	FRB+SVM	-	-	-	0.8886	-	0.6857
	Base+LCMine+CAEP	-	-	-	0.9714	-	-
	SMOTE+LCMine+CAEP	-	-	-	0.96	-	-
	SMOTE-ENN+LCMine+CAEP	-	-	-	0.9014	-	-
	SMOTE-TL+LCMine+CAEP	-	-	-	0.96	-	-
	ADASYN+LCMine+CAEP	-	-	-	0.9429	-	-
	Borderline-SMOTE+LCMine+CAEP	-	-	-	0.9771	-	-
	SafeãĂĂLevelãĂĂSMOTE+LCMine+CAEP			-	0.91	-	-
	ROS+LCMine+CAEP	-	-	-	0.9714	-	-
	ADOMS+LCMine+CAEP	-	-	-	0.8657	-	-
	SPIDER+LCMine+CAEP	-	-	-	0.9686	-	-
	AHC+LCMine+CAEP	-	-	-	0.9686	-	-
	SPIDER2+LCMine+CAEP	-	-	-	0.9743	-	-
	SMOTE-RSB+LCMine+CAEP	-	-	-	0.9571	-	-
	TL+LCMine+CAEP	-	-	-	0.9657	-	-
	CNN+LCMine+CAEP	-	-	-	0.9086	-	-
	RUS+LCMine+CAEP	-	-	-	0.9114	-	-
	OSS+LCMine+CAEP	-	-	-	0.9286	-	-
	CNNTL+LCMine+CAEP	-	-	-	0.8443	-	-
	NCL+LCMine+CAEP	-	-	-	0.9657	-	-
	SBC+LCMine+CAEP	-	-	-	0.5	-	-
	CPM+LCMine+CAEP	-	-	-	0.7843	-	-

TABLE 8. Experimental results of CFGVM and other existing algorithms on the *newthyroid*1, *shuttlec*2*vsc*4 dataset.

Dataset	Method	Accuracy	TPR	FPR	AUC	G-mean	F-measure
	Clustering-LMS	-	-	-	-	0.8629	-
	Clustering-SVD	-	-	-	-	0.8662	-
	CO^2RBFN-LMS	-	-	-	-	0.8471	-
	CO^2RBFN-SVD	-	-	-	-	0.8076	-
	Genetic-LMS	-	-	-	-	0.7652	-
	Genetic-SVD	-	-	-	-	0.7359	-
	Incremetal-LMS	-	-	-	-	0.7562	-
	Incremetal-SVD	-	-	-	-	0.7674	-
	CFGVM3	**1**	**1**	**0**	**1**	**1**	**1**
newthyroid1	IMCStacking	0.9972	1	0.56	-	-	-
	FLCStacking	0.9972	1	0.56	-	-	-
	FCStacking	0.9944	1	1.11	-	-	-
	LCStacking	0.9972	1	0.56	-	-	-
	CLR	0.9944	1	1.11	-	-	-
	RUSBagging-RF	0.9833	1	3.33	-	-	-
	RUSBoosting-RF	0.9972	1	0.56	-	-	-
	EasyEnsemble	0.9548	0.9833	7.38	-	-	-
	Base+LCMine+CAEP	-	-	-	0.9687	-	-
	SMOTE+LCMine+CAEP	-	-	-	0.9833	-	-
	SMOTE-ENN+LCMine+CAEP	-	-	-	0.9663	-	-
	SMOTE-TL+LCMine+CAEP	-	-	-	0.9722	-	-
	ADASYN+LCMine+CAEP	-	-	-	0.9861	-	-
	Borderline-SMOTE+LCMine+CAEP	-	-	-	0.9631	-	-
	SafeLevelSMOTE+LCMine+CAEP		-	-	0.9639	-	-
	ROS+LCMine+CAEP	-	-	-	0.9746	-	-
	ADOMS+LCMine+CAEP	-	-	-	0.9746	-	-
	SPIDER+LCMine+CAEP	-	-	-	0.9659	-	-
	AHC+LCMine+CAEP	-	-	-	0.9774	-	-
	SPIDER2+LCMine+CAEP	-	-	-	0.9631	-	-
	SMOTE-RSB+LCMine+CAEP	-	-	-	0.9663	-	-
	TL+LCMine+CAEP	-	-	-	0.9857	-	-
	CNN+LCMine+CAEP	-	-	-	0.9802	-	-
	RUS+LCMine+CAEP	-	-	-	0.9889	-	-
	OSS+LCMine+CAEP	-	-	-	0.9774	-	-
	CNNTL+LCMine+CAEP	-	-	-	0.9667	-	-
	NCL+LCMine+CAEP	-	-	-	0.9687	-	-
	SBC+LCMine+CAEP	-	-	-	0.5	-	-
	CPM+LCMine+CAEP	-	-	-	0.8893	-	-
	Clustering-LMS	-	-	-	-	0.9756	-
	Clustering-SVD	-	-	-	-	0.9696	-
	CO^2RBFN-LMS	-	-	-	-	0.9754	-
	CO^2RBFN-SVD	-	-	-	-	0.9841	-
	Genetic-LMS	-	-	-	-	0.9804	-
	Genetic-SVD	-	-	-	-	0.9769	-
	Incremetal-LMS	-	-	-	-	0.9179	-
	Incremetal-SVD	-	-	-	-	0.9615	-
	CFGVM3	**1**	**1**	**0**	**1**	**1**	**1**
shuttlec2vsc4	SVM	-	-	-	0.7	-	0.4
	SMOTE+SVM	-	-	-	0.9548	-	0.7152
	ADASYN+SVM	-	-	-	0.9548	-	0.7152
	sTL+SVM	-	-	-	0.9548	-	0.7152
	sSafe+SVM	-	-	-	0.9548	-	0.7152
	sRST+SVM	-	-	-	0.959	-	0.7288
	sCHC+SVM	-	-	-	0.944	-	0.6103
	EUSCHC+SVM	-	-	-	0.6957	-	0.1593
	AHC+SVM	-	-	-	0.7	-	0.4
	FRB+CHC+SVM	-	-	-	0.9493	-	0.6126
	FRB+SVM	-	-	-	0.9632	-	0.7395
	Clustering-LMS	-	-	-	-	0.9025	-
	Clustering-SVD	-	-	-	-	0.9007	-
	CO^2RBFN-LMS	-	-	-	-	0.9951	-
	CO^2RBFN-SVD	-	-	-	-	0.995	-
	Genetic-LMS	-	-	-	-	0.9551	-
	Genetic-SVD	-	-	-	-	0.9551	-
	Incremetal-LMS	-	-	-	-	0.9406	-
	Incremetal-SVD	-	-	-	-	0.9575	-
	CFGVM3	**1**	**1**	**0**	**1**	**1**	**1**

TABLE 9. Experimental results of CFGVM and other existing algorithms on the *ecoil*034*vs*5 dataset.

Dataset	Method	Accuracy	TPR	FPR	AUC	G-mean	F-measure
ecoil034vs5	SVM	-	-	-	0.4972	-	0
	SMOTE+SVM	-	-	-	0.7069	-	0.5629
	ADASYN+SVM	-	-	-	0.5889	-	0.2667
	sTL+SVM	-	-	-	0.7236	-	0.5901
	sSafe+SVM	-	-	-	0.7047	-	0.5578
	sRST+SVM	-	-	-	0.6799	-	0.5007
	sCHC+SVM	-	-	-	0.6747	-	0.5054
	EUSCHC+SVM	-	-	-	0.8111	-	0.6591
	AHC+SVM	-	-	-	0.5972	-	0.3111
	FRB+CHC+SVM	-	-	-	0.8217	-	0.5829
	FRB+SVM	-	-	-	0.8472	-	0.6337
	Base+LCMine+CAEP	-	-	-	0.9111	-	-
	SMOTE+LCMine+CAEP	-	-	-	0.9194	-	-
	SMOTE-ENN+LCMine+CAEP	-	-	-	0.8889	-	-
	SMOTE-TL+LCMine+CAEP	-	-	-	0.9667	-	-
	ADASYN+LCMine+CAEP	-	-	-	0.8806	-	-
	Borderline-SMOTE+LCMine+CAEP	-	-	-	0.8694	-	-
	SafeãĂĂLevelãĂĂSMOTE+LCMine+CAEP			-	0.9139	-	-
	ROS+LCMine+CAEP	-	-	-	0.9167	-	-
	ADOMS+LCMine+CAEP	-	-	-	0.9222	-	-
	SPIDER+LCMine+CAEP	-	-	-	0.8667	-	-
	AHC+LCMine+CAEP	-	-	-	0.9	-	-
	SPIDER2+LCMine+CAEP	-	-	-	0.8833	-	-
	SMOTE-RSB+LCMine+CAEP	-	-	-	0.8917	-	-
	TL+LCMine+CAEP	-	-	-	0.9333	-	-
	CNN+LCMine+CAEP	-	-	-	0.9028	-	-
	RUS+LCMine+CAEP	-	-	-	0.9556	-	-
	OSS+LCMine+CAEP	-	-	-	0.95	-	-
	CNNTL+LCMine+CAEP	-	-	-	0.9139	-	-
	NCL+LCMine+CAEP	-	-	-	0.9111	-	-
	SBC+LCMine+CAEP	-	-	-	0.5	-	-
	CPM+LCMine+CAEP	-	-	-	0.9333	-	-
	CFGVM3	**0.96**	**1**	**0.0444**	**0.9778**	**0.9773**	**0.8444**

6) The proposed algorithm CFGVM3 is more advanced than GVM, RBGVM, BBGVM, CGVM1, CGVM 2, CGVM3, CFGVM1, CFGVM2, A-SUWO, IMCStacking, SMOTE-IPF, FRB+CHC+SVM, LCMine+CAEP, CO2RBFN-LMS and CO2RBFN-SVD.

V. DISCUSSION

Cost-sensitive learning algorithms and feature selection algorithms can improve the class imbalance problem [48].The cost-sensitive learning algorithms improve the classification performance of imbalanced classification by assigning different misclassification costs to the minority and the majority class samples. The feature selection algorithms improve the classification performance of imbalanced classification by selecting some of the most representative features from the original features [48].

The GVM algorithm is a new classification algorithm proposed by Hong Zhao in 2016 [3]. Since it contains the design risk minimization and Monte Carlo algorithm, it has strong generalization ability and has been successfully applied in phishing detection [29], Android malware detection [30], groundwater status forecasting [31], electricity demand prediction [32]. However, it does not work well for the imbalanced classification. The BALO algorithm is a new nature-inspired algorithm proposed by E. Emary in 2016 [4]. Experiments have shown that the proposed BALO is better than particle swarm optimizer (PSO), genetic algorithms (GAs), binary bat algorithm (BBA), and the ALO for feature selection on 21 data sets.

To the best of our knowledge, our proposed method is the first research work aiming at utilizing the GVM and BALO algorithm to solve the imbalanced classification problem. The proposed algorithm CFGVM (CFGVM3) combines a cost-sensitive learning method and feature selection method based on BALO and GVM. Firstly, The BALO algorithm is used to improve the GVM algorithm to a cost-sensitive CGVM algorithm (Algorithm 1), and then the BALO algorithm is used to search the optimal feature subset to further improve the classification performance (Algorithm 2). In the GVM algorithm, we adopt the Monte Carlo training algorithm. In other words, we only randomly change one weight of one parameter matrix (the weight matrix between the hidden node and the input node, the bias of the hidden layer or the transfer function coefficient) in a small range every time, and GVM will accept the weight change while the overall cost is reduced. In this case, the output is insensitive to the small change of the input.

On the other hand, the GVM algorithm introduces the design risk to improve the generalization ability. The cost matrix is difficult to ascertain in the cost-sensitive learning algorithm. The BALO is a new metaheuristic optimization algorithm, which has advantages over other heuristic algorithms due to its adaptive boundary convergence and elitism mechanism. Therefore, the BALO algorithm is used to select the optimal cost matrix and the optimal feature in this paper. BALO is a binary variant of the ALO. In the BALO algorithm, each dimension of the search space is limited to 0 or 1. We propose a cost-sensitive algorithm CGVM (Algorithm 1) by setting different cost weight-based loss functions based on GVM. i represents the minority class, j represents the majority class, then C_{ij} represents the cost weight of misclassification of minority class samples, and the cost weight of misclassification of majority class samples represented by C_{ji}. Since we assume that the value of C_{ji} is 1, it is clear that the value of C_{ij} should be greater than 1, and we use the BALO algorithm to choose the optimal C_{ij} value. The position of the antlion in the BALO algorithm represents the value of C_{ij}. That is, when using BALO to select the cost weights of the minority class, the position of antlion represents the cost weights to be optimized. In Algorithm 2, the position of the antlion in the BALO algorithm represents the number of features. When the value of the antlion position is 1, it means that the feature is selected; but, when the value of the antlion position is 0, it means that the feature is not selected. The more significant feature can be achieved through the BALO algorithm.

Extensive experiments are carried out using eleven benchmark imbalanced datasets, which demonstrate that the proposed CFGVM algorithm can significantly improve the classification of imbalanced data sets compared to eight existing and seven recently published algorithms. Nine similar comparison algorithm contain GVM classification algorithm, the hybrid algorithm RBGVM with imbalance rate as cost weight, the hybrid algorithm BBGVM of choosing cost weight and feature with BPSO, the cost-sensitive algorithm CGVM1, CGVM2, CGVM3, the hybrid algorithm CFGVM1, CFGVM2, CFGVM3 of choosing cost weight and feature with BALO. From the experiment results, we can conclude the following: 1) The cost-sensitive CGVM1, CGVM2, CGVM3 algorithms outperform the single classification algorithm GVM; 2) The feature selection algorithm can further improve the classification performance based on the cost-sensitive algorithm; 3) *G-mean* can better evaluate the classification performance of imbalanced classifications compared to *F-measure* and *Accuracy*; 4) BALO is better than BPSO used for feature selection and cost weight selection in the imbalance classification; 5) The cost weight selected by BALO is better than the imbalance rate. Furthermore, compared with other hybrid algorithms in other references, the results show that the algorithm CFGVM3 proposed in this paper is better, which shows that the hybrid algorithm proposed in this paper is more advanced.

VI. CONCLUSION

Imbalanced classification problems are widely existing in medical diagnosis, network intrusion detection, credit card illegal transaction detection and software defect detection. However, traditional classification algorithms are often unable to effectively deal with this kind of problem. This is because they are biased toward the majority class and leaving aside the minority class.

In this paper, a CFGVM (CFGVM3) is proposed for an imbalanced classification problem. The proposed algorithm CFGVM combines cost-sensitive learning method and feature selection method based on BALO and GVM. The GVM algorithm has the advantage of strong generalization ability. However, it does not work well for the imbalanced classification. The BALO algorithm has advantages over other heuristic algorithms due to its adaptive boundary convergence and elitism mechanism. Firstly, The BALO algorithm is used to improve the GVM algorithm to a cost-sensitive CGVM algorithm, Specifically, the BALO algorithm is used to select the optimal cost matrix based on GVM in the CGVM algorithm. Secondly, the BALO algorithm is used to search the optimal feature subset based on CGVM. Furthermore, We present *G-mean* that can be used to measure the performance of cost-sensitive learning algorithm and feature selection algorithm in the imbalanced classification. Compared with the previous existing algorithms, the algorithm CFGVM3 is more capable of improving the classification performance of imbalanced data sets.

When the number of samples is less than 700 and the dimension of samples is less than 20, the proposed algorithm can achieve convergence when the number of iterations is set to 15. However, when the number of samples is large and the dimension is high, the number of iterations should be increased according to the characteristics and the number of samples.

The proposed algorithm is significant to solve the class imbalance problem. For example, in cancer diagnosis, the whole dataset with minority and majority class representing having cancer or not. if cancer patients are misclassified as normal patients, they will miss the best treatment time and eventually lead to death. The proposed algorithm can improve the classification performance of minority class. It is important to detect early-stage cancer.

As for future work, we can further investigate from four aspects as follows: 1) To establish an ensemble algorithm based on GVM algorithm and BALO algorithm; 2) To construct a hybrid algorithm based on the undersampling algorithm and feature selection algorithm. The proposed hybrid algorithm combines the advantage of undersampling algorithm and feature selection algorithm, and may achieve a better performance; 3) Online learning often suffers from concept drift, we can further develop a new algorithm based on the CFGVM to deal with this problem; 4) For high-dimensional imbalanced small-sample datasets, we can

propose a new algorithm based on the CFGVM to solve this problem.

REFERENCES

[1] C. Thomas, "Improving intrusion detection for imbalanced network traffic," *Secur. Commun. Netw.*, vol. 6, no. 3, pp. 309–324, Mar. 2013.

[2] C. Zhang, K. C. Tan, H. Li, and G. S. Hong, "A cost-sensitive deep belief network for imbalanced classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 1, pp. 109–122, Jan. 2019.

[3] H. Zhao, "General vector machine," 2016, *arXiv:1602.03950*. [Online]. Available: https://arxiv.org/abs/1602.03950

[4] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary ant lion approaches for feature selection," *Neurocomputing*, vol. 213, pp. 54–65, Nov. 2016.

[5] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012.

[6] S. Maldonado, R. Weber, and F. Famili, "Feature selection for high-dimensional class-imbalanced data sets using support vector machines," *Inf. Sci.*, vol. 286, pp. 228–246, Dec. 2014.

[7] F. Cheng, J. Zhang, and C. Wen, "Cost-sensitive large margin distribution machine for classification of imbalanced data," *Pattern Recognit. Lett.*, vol. 80, no. C, pp. 107–112, Sep. 2016.

[8] D. Wu, Z. Wang, Y. Chen, and H. Zhao, "Mixed-kernel based weighted extreme learning machine for inertial sensor based human activity recognition with imbalanced dataset," *Neurocomputing*, vol. 190, pp. 35–49, May 2016.

[9] P. Phoungphol, Y. Zhang, and Y. Zhao, "Robust multiclass classification for learning from imbalanced biomedical data," *Tsinghua Sci. Technol.*, vol. 17, no. 6, pp. 619–628, Dec. 2012.

[10] J. F. Díez-Pastor, J. J. Rodríguez, C. García-Osorio, and L. I. Kuncheva, "Random balance: Ensembles of variable priors classifiers for imbalanced data," *Knowl.-Based Syst.*, vol. 85, nos. 2–3, pp. 96–111, Sep. 2015.

[11] B. Krawczyk, M. Woźniak, and G. Schaefer, "Cost-sensitive decision tree ensembles for effective imbalanced classification," *Appl. Soft Comput.*, vol. 14, no. 1, pp. 554–562, Jan. 2014.

[12] Y. Sahin, S. Bulkan, and E. Duman, "A cost-sensitive decision tree approach for fraud detection," *Expert Syst. Appl.*, vol. 40, no. 15, pp. 5916–5923, Nov. 2013.

[13] V. López, S. del Río, J. M. Benítez, and F. Herrera, "Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data," *Fuzzy Sets Syst.*, vol. 258, no. C, pp. 5–38, Jan. 2015.

[14] S. Vluymans, D. Sánchez Tarragó, Y. Saeys, C. Cornelis, and F. Herrera, "Fuzzy rough classifiers for class imbalanced multi-instance data," *Pattern Recognit.*, vol. 53, pp. 36–45, May 2016.

[15] S. Oh, M. Su Lee, and B.-T. Zhang, "Ensemble learning with active example selection for imbalanced biomedical data classification," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 8, no. 2, pp. 316–325, Mar. 2011.

[16] C. L. Castro and A. P. Braga, "Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 6, pp. 888–899, Jun. 2013.

[17] A. Ghazikhani, R. Monsefi, and H. Sadoghi Yazdi, "Online cost-sensitive neural network classifiers for non-stationary and imbalanced data streams," *Neural Comput. Appl.*, vol. 23, no. 5, pp. 1283–1295, Oct. 2013.

[18] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognit.*, vol. 40, no. 12, pp. 3358–3378, Dec. 2007.

[19] B. X. Wang and N. Japkowicz, "Boosting support vector machines for imbalanced data sets," *Knowl. Inf. Syst.*, vol. 25, no. 1, pp. 1–20, 2010.

[20] S. Ali, A. Majid, S. G. Javed, and M. Sattar, "Can-CSC-GBE: Developing cost-sensitive classifier with gentleboost ensemble for breast cancer classification using protein amino acids and imbalanced data," *Comput. Biol. Med.*, vol. 73, pp. 38–46, Jun. 2016.

[21] S. Datta and S. Das, "Near-Bayesian support vector machines for imbalanced data classification with equal or unequal misclassification costs," *Neural Netw.*, vol. 70, pp. 39–52, Oct. 2015.

[22] A. C. Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten, "Cost sensitive credit card fraud detection using Bayes minimum risk," in *Proc. 12th Int. Conf. Mach. Learn. Appl.*, vol. 1, Dec. 2013, pp. 333–338.

[23] Q. Kang, X. Chen, S. Li, and M. Zhou, "A noise-filtered under-sampling scheme for imbalanced classification," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4263–4274, Dec. 2017.

[24] Q. Kang, L. Shi, M. Zhou, X. Wang, Q. Wu, and Z. Wei, "A distance-based weighted undersampling scheme for support vector machines and its application to imbalanced classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4152–4165, Sep. 2018.

[25] H. Liu, M. Zhou, and Q. Liu, "An embedded feature selection method for imbalanced data classification," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 3, pp. 703–715, May 2019.

[26] W. Tang, Z. Ding, and M. Zhou, "A spammer identification method for class imbalanced weibo datasets," *IEEE Access*, vol. 7, pp. 29193–29201, 2019.

[27] S. Mirjalili, "The ant lion optimizer," *Adv. Eng. Softw.*, vol. 83, no. C, pp. 80–98, May 2015.

[28] S. Mirjalili, P. Jangir, and S. Saremi, "Multi-objective ant lion optimizer: A multi-objective optimization algorithm for solving engineering problems," *Int. J. Speech Technol.*, vol. 46, no. 1, pp. 79–95, Jan. 2017.

[29] F. Feng, Q. Zhou, Z. Shen, X. Yang, L. Han, and J. Wang, "The application of a novel neural network in the detection of phishing Websites," *J. Ambient Intell. Humanized Comput.*, pp. 1–15, Apr. 2018.

[30] Q. Zhou, F. Feng, Z. Shen, R. Zhou, M.-Y. Hsieh, and K.-C. Li, "A novel approach for mobile malware classification and detection in Android systems," *Multimedia Tools Appl.*, vol. 78, no. 3, pp. 3529–3552, Feb. 2019.

[31] Q. Zhou, H. Chen, H. Zhao, G. Zhang, J. Yong, and J. Shen, "A local field correlated and Monte Carlo based shallow neural network model for nonlinear time series prediction," *ICST Trans. Scalable Inf. Syst.*, vol. 3, no. 8, 2016, Art. no. 151634.

[32] B. Yong, Z. Xu, J. Shen, H. Chen, Y. Tian, and Q. Zhou, "Neural network model with Monte Carlo algorithm for electricity demand forecasting in Queensland," in *Proc. Australas. Comput. Sci. Week Multiconf. (ACSW)*. ACM, 2017, p. 47.

[33] B. Yong, F. Li, Q. Lv, J. Shen, and Q. Zhou, "Derivative-based acceleration of general vector machine," *Soft Comput.*, vol. 23, no. 3, pp. 987–995, Feb. 2019.

[34] B. Yong, L. Huang, F. Li, J. Shen, X. Wang, and Q. Zhou, "A research of Monte Carlo optimized neural network for electricity load forecast," *J. Supercomput.*, pp. 1–14, Mar. 2019, doi: 10.1007/s11227-019-02828-3.

[35] X. Yang, Q. Zhou, J. Wang, L. Han, F. Feng, R. Zhou, and K.-C. Li, "FPGA-based approximate calculation system of general vector machine," *Microelectron. J.*, vol. 86, pp. 87–96, Apr. 2019.

[36] P. Lim, C. K. Goh, and K. C. Tan, "Evolutionary cluster-based synthetic oversampling ensemble (ECO-ensemble) for imbalance learning," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2850–2861, Sep. 2017.

[37] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[38] A. Ghazikhani, H. S. Yazdi, and R. Monsefi, "Class imbalance handling using wrapper-based random oversampling," in *Proc. 20th Iranian Conf. Electr. Eng. (ICEE)*, May 2012, pp. 611–616.

[39] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," in *Proc. 5th Int. Workshop Comput. Intell. Appl.* Hiroshima, Japan: Hiroshima Univ., 2009, pp. 24–29.

[40] T. M. R. Mohammad and F. A. Thabtah. (2017). *UCI Machine Learning Repository*. Accessed: Dec. 12, 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[41] *Keel Datasets*. Accessed: Apr. 14, 2020. [Online]. Available: http://www.keel.es/

[42] I. Nekooeimehr and S. K. Lai-Yuen, "Adaptive semi-unsupervised weighted oversampling (A-SUWO) for imbalanced datasets," *Expert Syst. Appl.*, vol. 46, pp. 405–416, Mar. 2016.

[43] C. Cao and Z. Wang, "IMCStacking: Cost-sensitive stacking learning with feature inverse mapping for imbalanced problems," *Knowl.-Based Syst.*, vol. 150, pp. 27–37, Jun. 2018.

[44] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," *Inf. Sci.*, vol. 291, pp. 184–203, Jan. 2015.

[45] G. Y. Wong, F. H. F. Leung, and S.-H. Ling, "A hybrid evolutionary preprocessing method for imbalanced datasets," *Inf. Sci.*, vols. 454–455, pp. 161–177, Jul. 2018.

[46] O. Loyola-González, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. García-Borroto, "Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases," *Neurocomputing*, vol. 175, pp. 935–947, Jan. 2016.

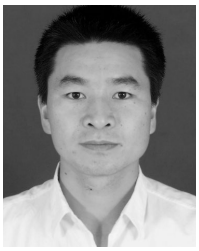
[47] M. D. Pérez-Godoy, A. J. Rivera, C. J. Carmona, and M. J. del Jesus, "Training algorithms for radial basis function networks to tackle learning processes with imbalanced data-sets," *Appl. Soft Comput.*, vol. 25, pp. 26–39, Dec. 2014.
[48] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, May 2017.

FANG FENG received the master's degree in computer science and technology from the Taiyuan University of Technology, in 2013, and the Ph.D. degree from the School of Information Science and Engineering, Lanzhou University, in 2019. She is also working at the School of Electronic and Information Engineering, Lanzhou Institute of Technology. Her research interests include machine learning, neural networks, and security.

KUAN-CHING LI (Senior Member, IEEE) is currently a Distinguished Professor at Providence University, Taiwan. His research interests include GPU/manycore computing, big data, and cloud. He is a member of the AAAS and a fellow of the IET. He was a recipient of awards and funding support from several agencies and industrial companies, and also received distinguished chair professorships from universities in China and other countries. He has been actively involved in many major conferences and workshops in program/general/steering conference chairman positions and member of the program committee and has organized numerous conferences related to high-performance computing and computational science and engineering. Besides publication in refereed journals and top conferences papers, he has coauthored/co-editor of several technical professional books published by CRC Press/Taylor & Francis, Springer, and McGraw-Hill.

JUN SHEN received the Ph.D. degree from Southeast University, China, in 2001. He is currently an Associate Professor with the University of Wollongong, Wollongong, NSW, Australia. He has authored over 190 articles in journals and conferences in computer science and information system area. His expertise is on computational intelligence and big data. He received the Outstanding Leadership Award from the IEEE Education Society. He has been an Editor, the PC Chair, a Guest Editor, and a PC Member for numerous journals and conferences published by the IEEE, ACM, Elsevier, and Springer.

QINGGUO ZHOU received the B.S. and M.S. degrees in physics and the Ph.D. degree in theoretical physics from Lanzhou University, in 1996, 2001, and 2005, respectively. He is currently a Professor with the School of Information Science and Engineering, Lanzhou University. His research interests include safety-critical systems, embedded systems, and real-time systems. He is also a Fellow of IET. He was a recipient of the IBM Real-Time Innovation Award, in 2007, the Google Faculty Award, in 2011, and the Google Faculty Research Award, in 2012.



XUHUI YANG received the Ph.D. degree from the School of Information Science and Engineering, Lanzhou University, in 2019. His current research interests include deep learning and the IoT.